FILE DAN DIREKTORI KHUSUS

8

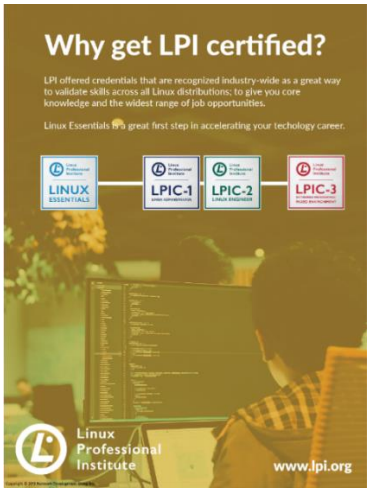
OBJEKTIF :

1. Mahasiswa Mampu Memahami Direktori dan File khusus pada sistem operasi Linux termasuk izin khusus.
2. Mahasiswa Mampu Memahami Istilah-Istilah Direktori dan File Khusus Pada Sistem Operasi Linux.

PENDAHULUAN

Dalam kebanyakan keadaan, izin dasar Linux yaitu: *read, write*, dan *execute*, sudah cukup untuk mengakomodasi kebutuhan keamanan masing-masing pengguna atau organisasi.

Namun, ketika beberapa pengguna perlu bekerja secara rutin pada direktori dan file yang sama, izin ini mungkin tidak cukup. Izin khusus *setuid*, *setgid* dan *sticky bit* dirancang untuk mengatasi masalah ini.



SETUID

Ketika izin *setuid* diatur pada file biner yang dapat dieksekusi (program), file biner dijalankan sebagai pemilik file, bukan sebagai pengguna yang mengeksekusinya. Izin ini ditetapkan pada beberapa utilitas sistem sehingga dapat dijalankan oleh pengguna normal, tetapi dijalankan dengan izin root, menyediakan akses ke file sistem yang biasanya tidak dapat diakses oleh pengguna normal.

Lihat skenario berikut di mana pengguna `sysadmin` mencoba untuk melihat konten file `/etc/shadow`:

```
sysadmin@localhost:~$ more /etc/shadow
/etc/shadow: Permission denied
sysadmin@localhost:~$ ls -l /etc/shadow
-rw-r-----. 1 root root 5195 Oct 21 19:57 /etc/shadow
```

Izin pada `/etc/shadow` tidak mengizinkan pengguna biasa untuk melihat (atau memodifikasi) file. Karena file dimiliki oleh pengguna *root*, administrator dapat mengubah izin sementara untuk melihat atau mengubah file ini.

Sekarang perhatikan perintah `passwd`. Ketika perintah ini dijalankan, ia memodifikasi file `/etc/shadow` , yang tampaknya tidak mungkin karena perintah lain yang dijalankan oleh pengguna `sysadmin` yang mencoba mengakses file ini gagal. Jadi, mengapa pengguna `sysadmin` dapat memodifikasi file `/etc/shadow` saat menjalankan perintah passwd ketika biasanya pengguna ini tidak memiliki akses ke file tersebut?

Perintah `passwd` memiliki izin *setuid* khusus. Ketika perintah passwd dijalankan, dan perintah mengakses file `/etc/shadow` , sistem bertindak seolah-olah pengguna yang mengakses file adalah pemilik perintah `passwd` (pengguna *root*), bukan pengguna yang menjalankan perintah.

Anda dapat melihat perizinan ini ditetapkan dengan menjalankan perintah `ls-l`:

```
sysadmin@localhost:~$ ls -l /usr/bin/passwd

-rwsr-xr-x 1 root root 31768 Jan 28 2010 /usr/bin/passwd
```

Perhatikan output dari perintah `ls` di atas; izin setuid diwakili oleh s di izin pemilik di mana izin eksekusi biasanya akan diwakili. Huruf kecil s berarti baik setuid dan izin eksekusi disetel, sedangkan huruf besar S berarti hanya setuid dan bukan izin eksekusi pengguna yang disetel.

Seperti izin *read, write*, dan *execute*, izin khusus dapat diatur dengan perintah chmod, menggunakan metode simbolis dan oktal.

Untuk menambahkan izin setuid secara simbolis, jalankan:

```
chmod u+s file
```

Untuk menambahkan izin setuid secara numerik, tambahkan 4000 ke izin file yang ada (asumsikan file aslinya memiliki 775 untuk izinnya dalam contoh berikut):

```
chmod 4775 file
```

Untuk menghapus izin setuid secara simbolis, jalankan:

```
chmod u-s file
```

Untuk menghapus izin setuid secara numerik, kurangi 4000 dari izin file yang sudah ada:

```
chmod 0775 file
```

Sebelumnya, kami telah menetapkan izin dengan metode oktal menggunakan kode tiga digit. Ketika kode tiga digit diberikan, perintah `chmod`

mengasumsikan bahwa digit pertama sebelum kode tiga digit adalah 0. Hanya jika empat digit ditentukan adalah perizinan khusus.

Jika tiga digit ditentukan saat mengubah izin pada file yang sudah memiliki izin khusus, digit pertama akan diatur ke 0, dan izin khusus akan dihapus dari file.

SETGID

Izin *setgid* mirip dengan *setuid*, tetapi menggunakan izin pemilik grup. Ada dua bentuk izin *setgid*: *setgid* pada file dan *setgid* pada direktori. Perilaku *setgid* bergantung pada apakah ia disetel pada file atau direktori.

SETGID PADA FILE

Izin *setgid* pada file sangat mirip dengan *setuid*; itu memungkinkan pengguna untuk menjalankan file biner yang dapat dieksekusi dengan cara yang memberi mereka akses grup tambahan (sementara). Sistem memungkinkan pengguna yang menjalankan perintah untuk secara efektif menjadi bagian dari grup yang memiliki file tersebut, tetapi hanya dalam program *setgid*.

Contoh yang bagus dari izin *setgid* pada file yang dapat dieksekusi adalah perintah `/usr/bin/wall` . Perhatikan izin untuk file ini serta pemilik grup:

```
sysadmin@localhost:~$ ls -l /usr/bin/wall
-rwxr-sr-x 1 root tty 30800 May 16  2018 /usr/bin/wall
```

Anda dapat melihat bahwa file ini di *setgid* dengan adanya `s` di posisi eksekusi grup. Karena eksekusi ini dimiliki oleh grup tty, ketika pengguna menjalankan perintah ini, perintah ini dapat mengakses file yang dimiliki grup oleh grup tty.

Akses ini penting karena perintah `/usr/bin/wall` mengirim pesan ke terminal, yang dilakukan dengan menulis data ke file seperti berikut:

```
sysadmin@localhost:~$ ls -l /dev/tty?
crw--w----. 1 root tty  4, 0 Mar 29  2013 /dev/tty0
crw--w----. 1 root tty  4, 1 Oct 21 19:57 /dev/tty1
```

Perhatikan bahwa grup tty memiliki izin menulis pada file di atas sementara pengguna yang tidak berada di grup tty ("orang lain") tidak memiliki izin pada file ini. Tanpa izin setgid, perintah `/usr/bin/wall` akan gagal.

SETGID PADA DIREKTORI

Saat disetel di direktori, izin *setgid* menyebabkan file yang dibuat di direktori menjadi milik grup yang memiliki direktori secara otomatis. Perilaku ini bertentangan dengan cara kepemilikan grup file baru biasanya berfungsi, karena secara default file baru adalah grup yang dimiliki oleh grup utama pengguna yang membuat file.

Selain itu, setiap direktori yang dibuat dalam direktori dengan izin setgid tidak hanya dimiliki oleh grup yang memiliki direktori setgid, tetapi direktori baru secara otomatis memiliki setgid di dalamnya juga. Dengan kata lain, jika sebuah direktori adalah setgid, maka setiap direktori yang dibuat dalam direktori tersebut mewarisi izin *setgid*.

Secara default ketika perintah `ls` dijalankan pada direktori, ia mengeluarkan informasi pada file yang ada di dalam direktori. Untuk melihat informasi tentang direktori itu sendiri tambahkan opsi `-d`. Digunakan dengan opsi `-l`, ini bisa digunakan untuk menentukan apakah izin setgid disetel. Contoh berikut menunjukkan bahwa direktori `/tmp/data` memiliki set izin setgid dan dimiliki oleh grup demo.

```
sysadmin@localhost:~$ ls -ld /tmp/data
```

```
drwxrwsrwx. 2 root demo 4096 Oct 30 23:20 /tmp/data
```

Dalam daftar, izin *setgid* diwakili oleh s di posisi eksekusi grup. Huruf kecil s berarti izin setgid dan eksekusi grup disetel:

```
drwxrwsrwx. 2 root demo 4096 Oct 30 23:20 /tmp/data
```

Huruf besar S berarti hanya setgid dan bukan izin eksekusi grup yang disetel. Jika Anda melihat huruf besar S di posisi eksekusi grup dari izin, maka ini menunjukkan bahwa meskipun izin *setgid* disetel, itu tidak benar-benar berlaku karena grup tidak memiliki izin eksekusi untuk menggunakannya:

```
drwxrwSr-x. 2 root root 5036 Oct 30 23:22 /tmp/data2
```

Biasanya file yang dibuat oleh `sysadmin` pengguna dimiliki oleh grup utama mereka, juga disebut `sysadmin`.

```
sysadmin@localhost:~$ id

uid=500(sysadmin) gid=500(sysadmin)
groups=500(sysadmin),10001(research),10002(development)
context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

Informasi UID dan GID pada contoh di atas mungkin berbeda dari output di virtual environment kita.

Namun, jika pengguna `sysadmin` membuat file di direktori `/tmp/data` , direktori *setgid* dari contoh sebelumnya, kepemilikan grup dari file tersebut bukanlah grup `sysadmin`, melainkan grup yang memiliki direktori demo:

```
sysadmin@localhost:~$  touch /tmp/data/file.txt

sysadmin@localhost:~$  ls -ld /tmp/data/file.txt
```

```
-rw-rw-r--. 1 bob demo 0 Oct 30 23:21 /tmp/data/file.txt
```

Mengapa seorang administrator ingin mengatur direktori *setgid*? Pertama, pertimbangkan akun pengguna berikut:

- User bob adalah anggota grup payroll.
- User sue adalah anggota grup staff.
- User tim adalah anggota grup acct.

Dalam skenario ini, ketiga pengguna ini perlu mengerjakan proyek bersama. Mereka mendekati administrator untuk meminta direktori bersama tempat mereka dapat bekerja sama, tetapi tidak ada orang lain yang dapat mengakses file mereka. Administrator melakukan hal berikut:

1. Membuat grup baru yang disebut team.

2. Menambahkan bob, sue, dan tim ke grup team.

3. Membuat direktori baru bernama `/home/team`.

4. Menjadikan pemilik grup dari direktori `/home/team` menjadi grup tim.

5. Memberi direktori `/home/team` izin berikut: rwxrwx---

Hasilnya, bob, sue, dan tim dapat mengakses direktori `/home/team` dan menambahkan file. Namun, ada kemungkinan masalah: ketika bob membuat file di direktori `/home/team` , file baru tersebut dimiliki oleh grup utamanya:

```
-rw-r-----. 1 bob payroll 100 Oct 30 23:21 /home/team/file.txt
```

Sayangnya, meskipun sue dan tim dapat mengakses direktori `/home/team` , mereka tidak dapat melakukan apapun dengan file bob. Izin mereka untuk file itu adalah izin orang lain (---).

Jika administrator menyetel izin setgid ke direktori `/home/team` , maka ketika bob membuat file, itu adalah milik grup team:

```
-rw-r-----. 1 bob team 100 Oct 30 23:21 /home/team/file.txt
```

Akibatnya, sue dan tim akan memiliki akses ke file melalui izin pemilik grup (r--).

Tentu saja, bob dapat mengubah kepemilikan grup atau hak akses lainnya setelah membuat file, tetapi itu akan menjadi membosankan jika ada banyak file baru yang dibuat. Izin *setgid* membuatnya lebih mudah untuk situasi ini.

PENGATURAN SETGID

Gunakan sintaks berikut untuk menambahkan izin *setgid* secara simbolis:

```
chmod g+s <file|directory>
```

Untuk menambahkan izin setgid secara numerik, tambahkan 2000 ke izin file yang ada (asumsikan dalam contoh berikut bahwa direktori aslinya memiliki 775 untuk izinnya):

```
chmod 2775 <file|directory>
```

Untuk menghapus izin setgid secara simbolis, jalankan:

```
chmod g-s <file|directory>
```

Untuk menghapus izin setgid secara numerik, kurangi 2000 dari izin file yang sudah ada:

```
chmod 0775 <file|directory>
```

STICKY BIT

Izin *sticky bit* digunakan untuk mencegah pengguna lain menghapus file yang tidak mereka miliki di direktori bersama. Ingatlah bahwa setiap pengguna dengan izin menulis pada direktori dapat membuat file di direktori itu, serta menghapus file apa pun di direktori, bahkan jika mereka bukan pemilik file tersebut!

Izin *sticky bit* memungkinkan file untuk dibagikan dengan pengguna lain, dengan mengubah izin tulis pada direktori sehingga pengguna masih dapat menambah dan menghapus file di direktori, tetapi file hanya dapat dihapus oleh pemilik file atau pengguna *root* .

Contoh yang baik dari penggunaan direktori *bit sticky* adalah direktori `/tmp` dan `/var/tmp` . Direktori ini dirancang sebagai lokasi di mana setiap pengguna dapat membuat file sementara.

Karena direktori ini dimaksudkan agar dapat ditulis oleh semua pengguna, mereka dikonfigurasi untuk menggunakan *sticky bit.* Tanpa izin khusus ini, pengguna akan dapat menghapus file apa pun di direktori ini, termasuk file milik pengguna lain.

Output dari perintah `ls -l` menampilkan sticky bit dengan karakter t di bit eksekusi dari grup izin lainnya:

```
sysadmin@localhost:~$ ls -ld /tmp

drwxrwxrwt 1 root root 4096 Mar 14  2016 /tmp
```

Huruf kecil t berarti *sticky bit* dan izin *execute* disetel untuk orang lain. Huruf besar T berarti hanya izin sticky bit yang disetel.

Sementara huruf besar S menunjukkan masalah dengan izin *setuid* atau *setgid*, huruf besar T tidak selalu menunjukkan masalah, selama pemilik grup masih memiliki izin eksekusi.

Untuk menyetel izin sticky bit secara simbolis, jalankan perintah seperti berikut:

```
chmod o+t <directory>
```

Untuk menyetel izin *sticky bit* secara numerik, tambahkan 1000 ke izin direktori yang ada (asumsikan direktori dalam contoh berikut ini awalnya memiliki 775 untuk izinnya):

```
chmod 1775 <file|directory>
```

Untuk menghapus izin *sticky bit* secara simbolis, jalankan:

```
chmod o-t <directory>
```

Untuk menghapus izin *sticky bit* secara numerik, kurangi 1000 dari izin direktori yang ada:

```
chmod 0775 <directory>
```

TAUTAN

Pertimbangkan skenario di mana ada file yang terkubur di sistem file yang disebut:

```
/usr/share/doc/superbigsoftwarepackage/data/2013/october/tenth/valuable-information.txt
```

Pengguna lain secara rutin memperbarui file ini, dan Anda perlu mengaksesnya secara teratur. Nama file yang panjang bukanlah pilihan yang ideal bagi Anda untuk mengetik, tetapi file tersebut harus berada di lokasi ini. Itu juga sering diperbarui, jadi Anda tidak bisa begitu saja membuat salinan file.

Dalam situasi seperti ini, tautan berguna. Anda dapat membuat file yang ditautkan ke file yang terkubur dalam. File baru ini dapat ditempatkan di direktori home atau lokasi lain yang nyaman. Ketika Anda mengakses file yang ditautkan, itu mengakses konten dari file *valuable-information.txt.*

Setiap metode penautan, keras dan simbolis, menghasilkan akses keseluruhan yang sama, tetapi menggunakan teknik yang berbeda. Ada pro dan kontra untuk setiap metode, jadi mengetahui kedua teknik tersebut dan kapan menggunakannya adalah penting.

MEMBUAT TAUTAN KERAS

Untuk memahami *hard links*, ada baiknya untuk memahami sedikit tentang bagaimana sistem file melacak file. Untuk setiap file yang dibuat, ada blok data di sistem file yang menyimpan metadata file. Metadata menyertakan informasi tentang file seperti izin, kepemilikan, dan stempel waktu. Metadata tidak menyertakan nama file atau konten file, tetapi mencakup hampir semua informasi lain tentang file.

Metadata ini disebut file *inode table*. Inode table juga menyertakan pointer ke blok lain pada sistem file yang disebut blok data tempat data disimpan.

Setiap file di partisi memiliki nomor identifikasi unik yang disebut nomor inode. Perintah `ls -i` menampilkan nomor inode file.

```
sysadmin@localhost:~$ ls -i /tmp/file.txt

215220874 /tmp/file.txt
```

Seperti pengguna dan grup, yang mendefinisikan file bukanlah namanya, melainkan nomor yang telah ditetapkan. Tabel inode tidak menyertakan nama

file. Untuk setiap file, ada juga entri yang disimpan di area data direktori (blok data) yang menyertakan asosiasi antara nomor inode dan nama file.

Di blok data untuk direktori `/etc` , akan ada daftar semua file di direktori ini dan nomor inode yang sesuai. Sebagai contoh:

File Name	Inode Number
passwd	123
shadow	175
group	144
gshadow	897

Saat Anda mencoba mengakses file `/etc/passwd` , sistem menggunakan tabel ini untuk menerjemahkan nama file menjadi nomor inode. Kemudian mengambil data file dengan melihat informasi di tabel inode untuk file tersebut.

Hard links adalah dua nama file yang mengarah ke *inode* yang sama. Misalnya, pertimbangkan entri direktori berikut:

File Name	Inode Number
passwd	123
mypasswd	123

File Name	Inode Number
shadow	175
group	144
gshadow	897

Karena file *passwd* dan *mypasswd* memiliki nomor *inode* yang sama, pada dasarnya keduanya adalah file yang sama. Anda dapat mengakses data file menggunakan salah satu nama file.

Saat Anda menjalankan perintah `ls -li`, nomor yang muncul untuk setiap file antara izin dan pemilik pengguna adalah nomor jumlah tautan:

```
sysadmin@localhost:~$ echo data > file.original
sysadmin@localhost:~$ ls -li file.*
278772 -rw-rw-r--. 1 sysadmin sysadmin 5 Oct 25 15:42 file.origina
l
```

Jumlah tautan menunjukkan berapa banyak *hard link* yang telah dibuat. Ketika nomor tersebut bernilai satu, maka file tersebut hanya memiliki satu nama yang ditautkan ke *inode*.

Untuk membuat *hard link*, perintah `ln` digunakan dengan dua argumen. Argumen pertama adalah nama file yang sudah ada untuk ditautkan, disebut target, dan argumen kedua adalah nama file baru untuk ditautkan ke target.

```
ln target link_name
```

Ketika perintah `ln` digunakan untuk membuat `hard link`, jumlah tautan bertambah satu untuk setiap nama file tambahan:

```
sysadmin@localhost:~$ ln file.original file.hard.1
sysadmin@localhost:~$ ls -li file.*
278772 -rw-rw-r--. 2 sysadmin sysadmin 5 Oct 25 15:53 file.hard.1
278772 -rw-rw-r--. 2 sysadmin sysadmin 5 Oct 25 15:53 file.origina
l
```

MEMBUAT LINK SIMBOL

Symbolic link, juga disebut *soft link*, hanyalah sebuah file yang mengarah ke file lain. Ada beberapa *symbolic link* yang sudah ada di sistem, termasuk beberapa di direktori `/etc` :

```
sysadmin@localhost:~$ ls -l /etc/grub.conf
lrwxrwxrwx. 1 root root 22 Feb 15  2011 /etc/grub.conf -> ../boot/
grub/grub.conf
```

Pada contoh diatas, file `/etc/grub.conf` "mengarah ke" file `../boot/grub/grub.conf`. Jadi, jika Anda mencoba untuk melihat konten dari file /etc/grub.conf , itu akan mengikuti penunjuk dan menampilkan konten dari file `../boot/grub/grub.conf` .

Untuk membuat *symbolic link*, gunakan opsi `-s` dengan perintah `ln`:

```
ln -s target link_name
```

```
sysadmin@localhost:~$  ln -s /etc/passwd mypasswd
```

```
sysadmin@localhost:~$  ls -l mypasswd

lrwxrwxrwx. 1 sysadmin sysadmin 11 Oct 31 13:17 mypasswd -> /etc/p
asswd
```

Perlu Diperhatikan!

Perhatikan bahwa bit pertama dari ls -l output adalah karakter l, yang menunjukkan bahwa tipe file adalah link.

MEMBANDINGKAN TAUTAN KERAS DAN SIMBOL

Sementara *hard* dan *symbolic links* memiliki hasil yang sama, teknik yang berbeda menghasilkan kelebihan dan kekurangan yang berbeda. Faktanya, keuntungan dari satu teknik mengkompensasi kerugian dari teknik lainnya.

Keuntungan: Hard Links tidak memiliki satu titik kegagalan.

Salah satu keuntungan menggunakan *hard link* adalah setiap nama file untuk konten file adalah setara. Jika Anda memiliki lima file yang ditautkan dengan keras, maka menghapus empat file ini tidak akan menghapus konten file yang sebenarnya.

Ingatlah bahwa file dikaitkan dengan nomor inode unik. Selama salah satu file *hard linked* tetap ada, maka nomor inode tersebut masih ada, dan data file masih ada.

Tautan simbolik, bagaimanapun, memiliki satu titik kegagalan: file asli. Pertimbangkan contoh berikut di mana akses ke data gagal jika file asli dihapus. File `mytest.txt` adalah tautan simbolis ke file `text.txt`:

```
sysadmin@localhost:~$ ls -l mytest.txt
lrwxrwxrwx. 1 sysadmin sysadmin 8 Oct 31 13:29 mytest.txt -> test.
txt
sysadmin@localhost:~$ more test.txt
hi there
sysadmin@localhost:~$ more mytest.txt
hi there
```

Jika file asli, file `test.txt` dihapus, maka file apa pun yang tertaut dengannya, termasuk file `mytest.txt`, gagal:

```
sysadmin@localhost:~$ rm test.txt
sysadmin@localhost:~$ more mytest.txt
mytest.txt: No such file or directory
sysadmin@localhost:~$ ls -l mytest.txt
lrwxrwxrwx. 1 sysadmin sysadmin 8 Oct 31 13:29 mytest.txt -> test.
txt
```

Keuntungan: Soft Links lebih mudah dilihat.

Terkadang sulit untuk mengetahui di mana terdapat *hard link* ke file. Jika Anda melihat file biasa dengan jumlah tautan yang lebih besar dari satu, Anda dapat menggunakan perintah `find` dengan kriteria pencarian `-inum` untuk menemukan file lain yang memiliki nomor *inode* yang sama. Untuk menemukan nomor *inode*, pertama-tama Anda akan menggunakan perintah `ls -i`:

```
sysadmin@localhost:~$ ls -i file.original
278772 file.original
sysadmin@localhost:~$ find / -inum 278772 2> /dev/null
/home/sysadmin/file.hard.1
```

```
/home/sysadmin/file.original
```

Soft Links jauh lebih visual, tidak memerlukan perintah tambahan di luar perintah `ls` untuk menentukan tautan:

```
sysadmin@localhost:~$ ls -l mypasswd

lrwxrwxrwx. 1 sysadmin sysadmin 11 Oct 31 13:17 mypasswd -> /etc/p
asswd
```

Keuntungan: Soft Links dapat ditautkan ke file apa pun.

Karena setiap sistem file (partisi) memiliki sekumpulan inode terpisah, hard link tidak dapat dibuat untuk mencoba melintasi sistem file:

```
sysadmin@localhost:~$ ln /boot/vmlinuz-2.6.32-358.6.1.el6.i686 Lin
ux.Kernel

ln: creating hard link `Linux.Kernel' => `/boot/vmlinuz-2.6.32-358
.6.1.el6.i686': Invalid cross-device link
```

Pada contoh sebelumnya, *hard link* telah dicoba untuk dibuat antara file di sistem file `/boot` dan sistem file `/;` gagal karena masing-masing sistem file ini memiliki satu set nomor inode unik yang tidak dapat digunakan di luar sistem file.

Namun, karena *simbolic link* menunjuk ke file lain menggunakan nama jalur, Anda dapat membuat soft link ke file di sistem file lain:

```
sysadmin@localhost:~$ ln -s /boot/vmlinuz-2.6.32-358.6.1.el6.i686
Linux.Kernel

sysadmin@localhost:~$ ls -l Linux.Kernel

lrwxrwxrwx. 1 sysadmin sysadmin 11 Oct 31 13:17 Linux.Kernel -> /b
oot/vmlinuz-2.6.32-358.6.1.el6.i686
```

Keuntungan: Soft link dapat ditautkan ke direktori.

Batasan lain dari *hard link* adalah tidak dapat dibuat di direktori. Alasan untuk batasan ini adalah karena sistem operasi itu sendiri menggunakan hard link untuk menentukan hierarki struktur direktori. Contoh berikut menunjukkan pesan kesalahan yang ditampilkan jika Anda mencoba melakukan *hard link* ke direktori:

```
sysadmin@localhost:~$ ln /bin binary
ln: `/bin': hard link not allowed for directory
```

Menautkan ke direktori menggunakan *simbolic link* diperbolehkan:

```
sysadmin@localhost:~$ ln -s /bin binary
sysadmin@localhost:~$ ls -l binary
lrwxrwxrwx. 1 sysadmin sysadmin 11 Oct 31 13:17 binary -> /bin
```